

UNITED STATES PATENT APPLICATION

FOR

**METHOD AND APPARATUS FOR ANONYMOUS SUBJECT-BASED
ADDRESSING**

INVENTOR:

Derek L. Collison

Prepared by:

Blakely, Sokoloff, Taylor & Zafman
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025
(408) 720-8598

Attorney's Docket No. 002982.P005

"Express Mail" mailing label number: EL 485757324US

Date of Deposit: December 21, 2000

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D. C. 20231

Shenise Ramdeen

(Typed or printed name of person mailing paper or fee)

Shenise Ramdeen

(Signature of person mailing paper or fee)

21 DEC 00

(Date signed)

**METHOD AND APPARATUS FOR ANONYMOUS SUBJECT-BASED
ADDRESSING**

[001] This is a continuation of U.S. Provisional Patent Application Number 5 60/171,930, entitled "Web Server and Web Server Method for Anonymous Subject-Based Addressing", filed December 22, 1999.

FIELD OF THE INVENTION

[002] The invention relates to network communications. More specifically, this 10 invention relates to a client computer accessing content from a remote server.

BACKGROUND OF THE INVENTION

[003] The World Wide Web and its ever-growing population and acceptance have developed a standard way of addressing information. Typically, the web browser 15 sends a file request to a host server that, in turn, "serves up" the file to the user.

[004] This is achieved using the Internet's underlying architecture, which is built on a foundation of HyperText Transfer Protocol (HTTP) or HyperText Transfer Protocol Secure (HTTPS), Transmission Control Protocol (TCP) and Internet Protocol (IP). In accordance with this set of protocols, a server program on a web server "listens" for a 20 client program (a client web browser) executing on a client computer to connect. The client browser connects to the server by utilizing a Uniform Resource Language (URL). A URL consists of a protocol specification (e.g. HTTP, FTP), a host destination, and a file specification. The host destination is effectively an address for point-to-point communications.

25 [005] This typical Internet communication is illustrated in Figure 1. In particular, **Figure 1** illustrates a system diagram of a browser, web server and application server of the prior art. Figure 1 includes web browser 11, Internet 13, server 15, servers

15.1-15.5, database 17 and content server 19. As shown server 15 includes files 16.1-
16.n and executable 18. Moreover, servers 15.1-15.5 can also include files 16.1-16.n and
executable 18 (not shown), which is described in more detail below. Web browser 11 is
coupled to servers 15 and 15.1-15.5 through Internet 13. In operation, web browser 11
5 submits a call through Internet 13 to one of servers 15 and 15.1-15.5. An example of
such a call could be “<http://www.rv.tibco.com/whitepaper.html>”. This call includes the
protocol specification (http), a host address (www.rv.tibco.com), which is the address of
server 15 and a file specification (whitepaper.html).

10 [006] Typically, server 15 processes this call as a file-based lookup. The
requested file is then returned by server 15 through Internet 13 to web browser 11 as an
HTTP compliant file, as, for example, a web page. Server 15 can return the file directly
to web browser 11 or process the requested file and return the results of the processing.
The processing of the file could, for example, result in a call to database 17 and/or an
accessing of the file from content server 19.

15 [007] Thus, the client, web browser 11, sets up a two-way or point-to-point
communication with server 15. The communication is established by web browser 11
sending an HTTP request to server 15. In the example of the call for
“<http://www.rv.tibco.com/whitepaper.html>”, server 15 (at “www.rv.tibco.com”)
interprets this request as a GET and determines that the client (web browser 11) wants a
20 file named “whitepaper.html”. Effectively, therefore, the call
“<http://www.rv.tibco.com/whitepaper.html>” is a file pointer where “whitepaper.html” is a
file stored on the server www.rv.tibco.com that is to be retrieved using the HTTP
protocol. Moreover, in the absence of the suffix “whitepaper.html”, the call
“<http://www.rv.tibco.com>” is a file pointer that is interpreted as a GET call of a default
25 page, typically “index.html” on the server “www.rv.tibco.com”. As illustrated by Figure
1, current Internet (HTTP/HTTPS/TCP/IP) architecture centers on a file-based web
server that is reliant on a point-to-point communication between the client and the server.

[008] File-based architectures combined with ever growing needs for expansion of 7-days-a-week-by-24-hours-a-day (7 x 24) services in light of higher security impose significant limitations on scalability and otherwise updating and modifying of the different servers. In particular, a file-based web server has to be updated internally to handle extensions to this system. This means shutting down the server for updates, upgrades, and scaling, thereby losing 7 x 24 availability. Thus the file-based server has become difficult to maintain, extend, and scale, especially as demands for expanded customer services and dynamic real-time content have increased.

[009] This problem is further exacerbated in multi-server environments, also illustrated in Figure 1: In multi-server environments as illustrated in Figure 1 having five servers 15.1-15.5, it is necessary to replicate the contents of server 15 onto each of servers 15.1-15.5. Accordingly, this means that copies of all the files 16.1-16.n and executable 18 must be replicated five times for each of servers 15.1-15.5. Similarly, as illustrated, the point-to-point links from servers 15 and 15.1-15.5 to database 17 and content server 19 must be made for each of such servers.

[010] As indicated before, updating and scaling requires internal updates to the web servers, which means that all the computers (servers) must be shut down, negatively impacting 7x24 availability. This also means that it is difficult for businesses to expand and adapt their systems to meet the needs of their customers. Accordingly, there is a need for an alternative system.

SUMMARY OF THE INVENTION

[011] One embodiment of the present invention provides a method that includes receiving, from a client computer, a point-to-point request message. The method also includes converting the point-to-point request message to a subject-based message. The subject-based message is also multicast. Additionally, a response is received to the subject-based message. The method also includes converting the response to the subject-based message to a point-to-point response message. Moreover, the point-to-point response message is transmitted back to the client computer.

10 BRIEF DESCRIPTION OF THE DRAWINGS

[012] Embodiments of the invention may be best understood by referring to the following description and accompanying drawings which illustrate such embodiments.

In the drawings:

Figure 1 illustrates a system diagram of a browser, web server and application server of the prior art;

Figure 2 illustrates a system diagram of a web browser, web server and application servers, according to embodiments of the present invention;

Figure 3 illustrates a system diagram of a browser, web server, application servers and the functionality therein, according to embodiments of the present invention;

Figure 4 is a flowchart illustrating a method for processing requests that incorporates subject-based messages, according to embodiments of the present invention; and

Figure 5 illustrates a block diagram of a system application of embodiments of the present invention.

DETAILED DESCRIPTION

[013] A method and an apparatus for anonymous subject-based addressing in network communications are described. Accordingly, different terminology related to subject-based addressing in a network is disclosed herein. The term “publish” is synonymous with the term “send”, while the term “subscribe” is synonymous with the term “listen.” Moreover, in the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced without these specific details.

[014] **Figure 2** illustrates a system diagram of a web browser, web server and application servers, according to embodiments of the present invention. In particular, Figure 2 includes web browser 11, network 202, server 204, network path 206 and application servers 208-214. Web browser 11 is coupled to server 204 through network 202. Further server 204 is coupled to application servers 208-214 through network path 206.

[015] In one embodiment, network 202 is a local area network (LAN). In another embodiment, network 202 is a wide area network (WAN). In one such embodiment, network 202 is the Internet. Further, network 202 can be a combination of different networks that provide communication among web browser 11 and servers 204-214. Additionally, the topology of Figure 2 is by way of example and not by way of limitation, as other types of topologies can be incorporated into embodiments of the present invention. For example, the coupling among servers 204-214 can be through network 202 instead of through network path 206 that is separate from network 202.

[016] **Figure 3** illustrates a system diagram of a browser, web server, application servers and the functionality therein, according to embodiments of the present invention. In particular, Figure 3 illustrates a method of operation for server 204 and application servers 208-214, according to embodiments of the present invention. For the

sake of simplicity, one block within Figure 3 is illustrating the functionality of each of application servers 208-214. In operation, web browser 11 transmits an HTTP protocol request for data to server 204 through network 202. Server 204 receives the HTTP request, at process block 302.

5 [017] Server 204 converts this HTTP request into a subject-based request for a publish/subscribe communication, at process block 304. Publish/subscribe communications technology expedites the delivery of real-time information in, for example, the financial industry. In particular, publish/subscribe communications enables applications in any environment to share up-to-date information reliably and
10 transparently. In such communications, a given publish/subscribe message traverses a network, thereby allowing devices, such as servers, that are connected to this network to subscribe to different publish/subscribe messages. For example, a given publish/subscribe message may be regarding the request for a particular file that is residing on different servers on the network. This message would be formatted such that
15 when the message is received the subscribers know which file needs to be transmitted back to the requesting device that originally transmitted the publish/subscribe message. In contrast, point-to-point communications would send the same message individually to each subscriber, thereby wasting network bandwidth and slowing delivery of such messages.

20 [018] Additionally, server 204 assigns a reply subject, at process block 306 and publishes the subject-based message, at process block 308. In an embodiment, a subject name is a character string that describes the content of a message and specifies the destination of a message. Subject-based addressing technology helps messages reach their destinations without involving application designers in the details of network
25 addresses, protocols, packets, ports and sockets. In one embodiment, such application designers devise conventions for intuitive, human-readable subject names.

[019] Moreover, traditional network programming incorporates IP addresses into

messaging distribution, thereby binding programs to specific computing devices. In contrast, subject-based addressing applications share information by subject names, thereby freeing applications to run on any computing device at any location on a network.

In particular, interested parties, such as a server on the network, listen for (subscribe to) specific subject names. Accordingly, when a first device (a publisher) publishes a message on a specific subject name and other devices on the network (subscribers) are listening for (subscribing to) that subject name, this subject-based addressing scheme reliably routes the messages from the publisher to the subscribers. Any application executing on a device on the network can, therefore, send messages to any other applications executing on devices on the network. This subject-based addressing scheme makes distributed systems much more flexible and maintainable.

[020] Applications, therefore, receive messages by listening. Additionally, in an embodiment, listening by a given device associates a subject name with a callback function which can be executed on such a device. In particular, when a message arrives, subject-based addressing software dispatches this message to the appropriate callback function by matching a given subject name to a particular callback function.

[021] As described, subject-based addressing technology allows for location transparency. Publishers and subscribers within this technology can run on any computer on a network. Further, server applications can migrate and replicate (to share a heavy client load) with no impact on existing clients. For example, because the publisher (or the client) is not required to know the specific address (e.g., IP address) for its subscribers, subscribers can be added, removed or modified without impacting how the publisher transmits its subject-based messages. Accordingly, this subject-based addressing technology allows end users to build scalable workflow systems that can adapt easily to change and growth.

[022] One example of building a scalable workflow system can occur during times of heavy load, as the system may need to respond by adding resources to

accommodate this load. Old systems would need to plan this upgrade to the system in advance. Even then one can be caught off guard at the sheer size of the Internet population, and the number of simultaneous requests that need to be processed by the web server. When this occurs, current architectures cannot scale, they are statically created systems with bounded resources, as quick adaptation is not an available option.

With embodiments of the present invention, additional resources can be added dynamically on the fly, and in real-time, to decrease latencies and avoid denial of service to customers. These backend systems can also take themselves off-line when the load subsides.

[023] Returning to Figure 3, in an embodiment, this subject-based request message is multicast over network 206. In one such embodiment, application servers 208, 210, 212 and 214 are listening for the requests for the given subject matter, at process block 310. Upon receipt of the subject-based request, application servers 208, 210, 212 and 214 process such a request by retrieving the content for the given subject matter, at process block 312. In one embodiment, the content is retrieved from a local database stored within application servers 208, 210, 212 and 214. However, embodiments of the present invention are not so limited, as such content can be retrieved from other locations. For example, the content can be stored on another remote server. Moreover, at process block 314, application servers 208, 210, 212 and 214 publish this retrieved content using a message having a reply subject which was assigned by server 204 at process block 306.

[024] In an embodiment, at process block 316, server 204 begins listening for a response to the subject-based message once the subject-based request is published at process block 308. Upon receipt of the response to the subject-based message from any of application servers 208, 210, 212 or 214, server 204 generates a point-to-point response based on the fields and content within the response. In an embodiment, this point-to-point response is based on the HTTP protocol. This generation of the point-to-

point response is described in more detail below in conjunction with Figure 4. Server 204 transmits this point-to-point response back to web browser 11 through network 202, at process block 318. In an embodiment, upon receipt of such a response, web browser 11 generates a web page displaying the contents of this response to a user of web browser 5 11. However, embodiments of the present invention are not so limited, as web browser 11 could perform other actions upon receipt of such a response. For example, this response could be a redirect to another web page, thereby causing web browser 11 to generate a new HTTP protocol request to server 204.

[025] Accordingly, web browser 11 sends an HTTP/HTTPS compliant protocol 10 through network 202 using a TCP/IP/HTTP/HTTPS set of protocols. Server 204 receives this call, packet or message from web browser 11 and converts it into a subject-based call. For example, an originated call, “<http://www.rv.tibco.com/whitepaper.html>”, from web browser 11 is converted into a subject-based message, “whitepaper.html”. Note, 15 server 204 receiving this browser call would be a server identified by address “www.rv.tibco.com”. After assigning of a reply subject, such as “A762 whitepaper.html”, server 204 multicasts the subject-based request over network 206, such that all of the servers that are listening for this particular subject-based message (e.g., application servers 208-214) can receive the message. In an embodiment, application servers 208-214 can be on a local or wide area network of multiple computers. In one 20 embodiment, application servers 208-214 are scalable. Moreover, in an embodiment, servers 204 and 208-214 can be located on a same server.

[026] **Figure 4** is a flowchart illustrating a method for processing requests that incorporates subject-based messages, according to embodiments of the present invention. In particular, Figure 4 illustrates method 400 that commences with a call that is initiated 25 at web browser 11, at process block 402. Web browser 11 initiates the call by sending an HTTP/HTTPS compliant call through network 202, using the TCP/IP/HTTP/HTTPS set of protocols, at process block 404. Accordingly, server 204 receives the call, at process

block 406. Server 204 maps the point-to-point call or request to a subject-based message, at process block 408. For example, if the call or request coming from web browser 11 is a file request, such as “<http://.../whitepaper.html>”, this call or request is mapped into the subject space of “whitepaper.html”.

5 [027] Moreover, the HTTP request is mapped to name/value pairs and packed into a subject-based message, at process block 410. Additionally, server 204 generates a reply subject, at process block 412. In one embodiment, this generation of a reply subject includes the generation of a listen event for all subsequent messages published on that particular subject by server 204. Such messages represent the response whose content is
10 to be sent to web browser 11. Moreover, the generation of a reply subject includes the publishing of the subject-based request.

15 [028] When one of application servers 208-214 returns a response to the subject-based request, server 204 receives this response based on the previously assigned reply subject, at process block 414. Further, server 204 generates a point-to-point response based on the contents and the type fields of this subject-based response, at process block 416. In an embodiment, the point-to-point response is an HTTP or HTTPS response. Server 204 then sends this point-to-point response back to web browser 11, at process block 418. In one embodiment, web browser 11 displays the response as an HTML web page, at process block 420. However, as described above, the response coming back
20 from server 204 is not limited to a web page for display, as embodiments of the present invention can incorporate any other type of response communication between a server and a client computer.

25 [029] **Figure 5** illustrates a block diagram of a system application of embodiments of the present invention. Similar to Figure 2, Figure 5 includes web browser 11, network 202, server 204, network path 206 and application servers 208-214. Web browser 11 is coupled to server 204 through network 202. Further server 204 is coupled to application servers 208-214 through network path 206.

[030] Additionally, as shown, application server 214 can include a number of servers therein (application servers 214.1-214.n). In one embodiment, application servers 214.1-214.n are connected through a local area network (LAN). In another embodiment, application servers 214.1-214.n are connected through a wide area network (WAN). In 5 one such embodiment, application servers 214.1-214.n are connected through the Internet. Further, application servers 214.1-214.n are connected through a combination of different networks that provide communication these application servers.

[031] Subject-based addressing, in contrast to point-to-point-based addressing, allows queuing and load balancing of request messages, which is illustrated by Figure 5.

10 Embodiments of the present invention also provide for the incorporation of distributed queues therein, which will be illustrated by the configuration and operations of application servers 214.

[032] In operation, application servers 214.1-214.n are associated with a distributed queue such that received subject-based messages, as described above in conjunction with Figures 3-4 are placed in this queue. Moreover, one of application 15 servers 214.1-214.n is designated as the active scheduler for this distributed queue. In an embodiment, the active scheduler is determined based on scheduler weight. Scheduler weight represents the ability of a member session to fulfill the role of scheduler, relative to other members of the same queue. In one such embodiment, the scheduler weight is 20 based on the availability of resources therein. Moreover, in an embodiment, application server 214.1-214.n having the highest scheduler weight is designated as the scheduler.

[033] Additionally, in one embodiment, the given server (e.g., application server 214.1), which is the active scheduler, sends heartbeat messages at specified intervals to the other server members of the distributed queue (e.g., application servers 214.2-214.n).

25 These heartbeat messages inform other member servers that the active scheduler is still alive. Accordingly, each of the servers of the distributed queue should specify the same scheduler activation interval for receiving these heartbeat messages. When the heartbeat

messaging from the active scheduler has been silent for this activation interval, the queue member of the remaining queue members having the greatest scheduler weight takes the active scheduler's place as the new active scheduler.

[034] Accordingly, this type of scheduling provides the necessary fault tolerance
5 such that a given member server 214.1-214.n acts as the scheduler for the distributed queue. Any member server 214.1-214.n has the potential to become the scheduler, as these fault tolerant parameters (e.g., scheduling weight) select the most suited member server as the active scheduler for the distributed queue.

[035] Moreover, all the application servers, including the active scheduler, that
10 are part of the distributed queue are defined to be workers or listeners, as these application servers listen for the subject-based messages coming into the distributed queue and are assigned the tasks of working on or processing these subject-based messages. In an embodiment, application servers 214.1-214.n of the distributed queue could share the same reusable correspondent name indicating that they are members of the queue with that name. In one such embodiment, each member of the distributed queue listens for the same subject. However, even when each member listens, for each inbound message, only one member processes the message.
15

[036] In particular, when a subject-based message is received into the distributed queue for application servers 214.1-214.n, the active scheduler (e.g., application server
20 214.1) assigns the task of processing this subject-based message to any of application servers that are associated with the distributed queue, including the active scheduler. In one embodiment, the active scheduler assigns the processing task for a given subject-based message based on worker or listener weights.

[037] In one such embodiment, the active scheduler assigns the processing task
25 to the available worker or listener with the greatest worker or listener weight. In an embodiment, a worker or listener is considered available unless (1) the pending tasks assigned to this worker or listener exceeds its task capacity or (2) the worker or listener is

the active scheduler. However, the active scheduler does assign tasks to itself when the other workers or listeners are not available.

[038] Task capacity is defined as the maximum number of tasks that a worker or listener can accept. In an embodiment, when the number of accepted tasks reaches this 5 maximum, the worker or listener cannot accept additional tasks until the completion of at least one of the currently pending tasks. Upon receipt of a task based on an incoming subject-based message, the active scheduler assigns the tasks to the worker or listener with the greatest worker or listener weight – unless the pending tasks assigned to such a worker or listener exceeds its task capacity. In one embodiment, when this preferred 10 worker or listener has exceeded its task capacity, the active scheduler assigns this new task to the worker or listener with the next greatest worker or listener weight.

[039] In one embodiment, a default task capacity assigned to a server when the server is associated with the distributed queue is one. However, this value can be modified based on a number of different factors. In an embodiment, on a server that 15 includes a number of processors that execute multi-threaded programs, such a server that devotes n threads and n processors to inbound tasks has a task capacity of n .

[040] In one embodiment, communication time lag is factored into the task capacity for a given server. In most distributed queue applications, the communication time is an insignificant fraction of the task turnaround time. In other words, the time 20 required to assign a task and signal its completion is very small in comparison to the time required to process the actual task. For example, when an average task turnaround time is 2000 milliseconds, wherein the communication time contributes 10 milliseconds to such a total, the task capacity is the same as the number of processors or threads.

[041] However, in certain situations communication time can be significant. For 25 example, communication time can become significant when the queue members are distributed at distant sites connected by a WAN or even the Internet or an Intranet. When communication time becomes significant, the meaning of task capacity changes. In

particular, instead of signifying the number of tasks that a listener can process concurrently, the task capacity signifies the number of tasks that can fill the listener's capacity despite the communication time lag. For example, when the average task turnaround time is 1500 milliseconds, of which the average task processing time and 5 communication time contribute 1000 milliseconds and 500 milliseconds, respectively, to the total, setting of the task capacity to account for the communication time minimizes the listener's idle time between tasks.

[042] Accordingly, when tuning task capacity to compensate for communication time lag, balance is critical. "Underloading" a listener (by setting its task capacity too low) can cause the listener to remain idle while waiting for the active scheduler to assign its next task. Conversely, "overloading" a listener (by setting its task capacity too high) can cause some assigned tasks to wait, while other listeners that might have accepted those tasks to remain idle. In an embodiment, application servers 214.1-214.n support multiple qualities of services for delivery of subject-based messages.

[043] As described, distributed queuing addresses the problem that it is undesirable for all n-application servers in making up a particular application server 214 to get every message published by server 204. Typically, all servers in the group 214 perform the same task. For example, it may be undesirable because each of the n-application servers in group 214 will take an action. In addition, if each application 15 server is to receive the message (and possibly respond to it) this will consume network bandwidth.

[044] Nonetheless, it is imperative for at least one of the n-application servers in group 214 to receive the message, and that request message be load balanced between all possible application servers i.e., ideally the system should deliver the message to only one 20 of the n-application servers 214. This type of situation arises where a large number n of application servers 214 is required to provide the desired level of fault tolerance and/or load balancing. Fault tolerance is important, for example, in situations where

applications may be unstable or message delivery is absolutely critical.

[045] In embodiments of the present invention, each application server in the group 214 sends messages to one of the n-application servers, say server 214.1, acting as the active scheduler, giving an indication of its weight. The active scheduler 214.1 then

5 responds by sending the received subject based message to the particular application server in the group with the greatest weight. Accordingly, a distributed queue of subscribing servers in the group 214 can accept messages that represent tasks (updates and queries). The system assigns each task to exactly one of the servers, while the group of servers and the distribution of tasks remain completely transparent to the web server.

10 The distributed queuing and task scheduling features are described in more detail in published PCT application WO 99/09490, which is hereby incorporated by reference.

[046] Additionally, the computing devices, such as the servers, described herein, can include processing units and memory (not shown). Such memory includes a machine-readable medium on which is stored a set of instructions (i.e., software) embodying any one, or all, of the methodologies described above. Software can reside, completely or at least partially, within this memory and/or within such a processing unit. For the purposes of this specification, the term "machine-readable medium" shall be taken to include any mechanism that provides (i.e., stores and/or transmits) information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium includes read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.); etc.

15

20

[047] Thus, a method and an apparatus for anonymous subject-based addressing in network communications have been described. Although the present invention has been described with reference to specific exemplary embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing

from the broader spirit and scope of the invention. For example, embodiments of the present invention are described that are employing TCP/IP/HTTP/HTTPS protocols between a given web browser and a given server. However, such embodiments are by way of example and not by way of limitation, as any other type of point-to-point protocol
5 can be received by a given server and converted to a subject-based message.

[048] Another example of a modification that can be made to these embodiments is the elimination of network path 206, as shown in Figures 2, 3 and 5. In particular, embodiments of the present invention are not limited to the transmission of a subject-based message to external servers coupled to network path 206, as a subject-
10 based message may be transmitted to processes locally within server 204. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.